疑似乱数生成器の安全性とモンテカルロ法 (確率数値解析に於ける諸問題,VI)

AUTHOR(S):

杉田, 洋

CITATION:

杉田, 洋. 疑似乱数生成器の安全性とモンテカルロ法 (確率数値解析に於ける諸問題,VI). 数理解析研究所講究録 2004, 1351: 33-40

ISSUE DATE:

2004-01

URL:

http://hdl.handle.net/2433/64973

数理解析研究所講究録 1351 巻 2004 年 33-40

疑似乱数生成器の安全性とモンテカルロ法

杉 田 洋　　(阪大理)

1. 序

小論では，疑似乱数生成器の安全性について述べる．それはすでに1980年代に計算機科学，とくに暗号理論において提案された概念であるが，ここで提示するのはその'モンテカルロ版' である．

そのためには，モンテカルロ法を再定義する必要がある．その際の基本的な考え方は,「ランダムとは何か，どのように生成するのか」ということを不問にして，コルモゴロフの確率論の精神に従ってモンテカルロ法を構成することである．といっても，実際に今日行われているモンテカルロ法の計算手順を変えるわけではない．その各々の手順が真に意味するところを数学的に明らかにする，ということである．

省みれば，我々はモンテカルロ法をきわめて直感的に扱い過ぎてきた．直感的に扱っても十分正しい答えを得られるし，非専門家にもそれなりに説明をすることができた．翻って，モンテカルロ法の新しい定義はずっと形式的 (= 数学的) で，とくに非専門家が理解するのは，コルモゴロフの確率論の場合と同様に，なかなか困難かも知れない．たとえば，確率変数が非専門家にとっては偶然に支配される変量であっても，専門家にとっては確率空間で定義された可測関数に過ぎないように，旧来のモンテカルロ法では疑似乱数生成器は何やらランダムな数列を作るサブルーチンであったのが，新しいモンテカルロ法の定義においては，それは短いビット列を長いビット列に写す関数に過ぎない (定義 1).

計算機科学で盛んに研究されている「計算量的に安全な疑似乱数生成器」(または暗号理論的に安全な疑似乱数生成器ともいう) は，理論上，最も汎用で最も完全な疑似乱数生成器である．ただし，計算量的に安全な疑似乱数生成器ではないかと予想される候補はいくつも提示されているが，その存在は厳密には未解決であり，また，具体的な確率論的性質がほとんど分かっていない．

しかし，もし疑似乱数の用途を限ってしまえば，その用途に関して安全な疑似乱数生成器は存在する可能性がある．計算量理論とは無関係に安全性を議論できるかも知れないからである．とくに用途を数値積分 (モンテカルロ積分) に限ってしまえば，— 実際，モンテカルロ計算のほとんどは数値積分であるが — それ専用の安全な疑似乱数生成器はすでに実現されている，ということができる．

2. モンテカルロ法の再定義

小論で扱う確率空間は，常に有限集合 Ω とそのベキ集合 2^Ω，そして Ω 上の一様確率測度，というタイプの確率空間であるとする．Ω の如何にかかわらずその上の一様確率測度は P で表し，確率変数の平均と分散は $\mathbf{E}$ と $\mathbf{Var}$ で表す．

モンテカルロ法ではある与えられた確率変数 X の一般的な (generic) 値を算出することが望まれる．しかしながら，一般には算出した値が X の例外的な値であるリスクがある．場合によっては X の例外値を避けるために有用な情報を計算者が持っているかもしれないが，ここでは簡単のためそのような情報は一切持っていないと仮定しよう．それで小論ではいつも一様確率測度を考えるのである．

モンテカルロ法における問題解決は，計算者が自由に X のサンプルを選択する権利を有すると同時に，その結果責任を負う，という了解の下でなされる．このとき，そのリスクの確率的評価をできるだけ正確に行うことが，モンテカルロ法の設計者にとって重大な責務となる．モンテカルロ法の目的は「どのようにすればよいサンプル (X の一般的な値) が得られるか」に答えることではなく，悪いサンプルを拾ってしまうリスクの確率的評価を得ることである．

モンテカルロ法を'賭け'になぞらえて，次の 2 つの例を考えてみよう．

例 1. ジョーカー 1 枚を含む 32 枚のよくシャッフルされたトランプカードの中から 1 枚選ぶ．もし，それがジョーカーでなければ勝ち，ジョーカーであれば負け，という賭けを考える．このとき負ける確率は 1/32 である．

例 2. ジョーカー 2^{10^6-5} 枚を含む 2^{10^6} 枚のよくシャッフルされたトランプカードの中から 1 枚選ぶ... これは如何にも非現実的であるから，同等の賭けを次のようにコンピュータで行うことを考えよう: 集合 $\{0,1\}^{10^6}$ の部分集合 A があって $\#A = 2^{10^6-5}$ とする．1 つの $\omega \in \{0,1\}^{10^6}$ を選んだとき，$\omega \notin A$ ならば勝ち，$\omega \in A$ ならば負け，という賭けを考える．このときも負ける確率は 1/32 である．

例 1 はとくに問題はないだろう．では，例 2 はどうか．例 2 は実際にモンテカルロ法でよく扱われている大規模な問題を単純化したものである．そしてモンテカルロ法の理論的解析は，通常，例 2 のように'賭け'のリスクをきちんと評価した時点で終わる．

しかし，例 2 の賭けは実践するときに現実的な問題が生じる．そもそも，一体どうやって 1 つの $\omega \in \{0,1\}^{10^6}$ をコンピュータに入力すればよいのか．人がキーボードから入力することは，もはや不可能であるから，何らかの道具が必要である．もし「物理乱数」を用いて ω の入力を肩代わりさせるとすると，そこで数学的な解析は終了してしまう．ここでは，小論の表題にある'疑似乱数生成器'を用いて ω の入力を肩代わりさせることを考える．

定義 1. $n < L$ のとき，関数 $g : \{0,1\}^n \to \{0,1\}^L$ を疑似乱数生成器 (pseudo-random generator) という．ここで g の入力 $\omega' \in \{0,1\}^n$ を種 (seed)，出力 $g(\omega') \in \{0,1\}^L$ を疑似乱数 (pseudo-random numbers, bits) という[1].

たとえばある疑似乱数生成器 $g : \{0,1\}^{100} \to \{0,1\}^{10^6}$ を用いて例 2 の賭けを実践するとは，1 つの種 $\omega' \in \{0,1\}^{100}$ をキーボードから入力し，疑似乱数 $g(\omega')$ を生成して，もし $g(\omega') \notin A$ ならば勝ち，$g(\omega') \in A$ ならば負け，と判定することを意味する．ここで，このように g を用いて実践した例 2 の賭けは数学的には別の新しい賭けであることに注意しなければならない．

では，この新しい賭けのリスク (負けの確率) $P(g(\omega') \in A)$ はどうなるか．特別な場合を除いて $P(g(\omega') \in A)$ を具体的に計算または評価することは難しい．それにもかかわらず，計算者は疑似乱数生成器を用いていても例 2 の賭けを実践しているつもりでいるから，当然，負けの確率は例 2 の場合と変わりのないこと，つまり $P(\omega \in A) = P(g(\omega') \in A)$ を期待している．言い換えれば，疑似乱数生成器 g はこれを成り立たせるようなものが望まれているのである．

注意 1. 疑似乱数生成器 $g : \{0,1\}^n \to \{0,1\}^L$ に 1 つの種 $\omega' \in \{0,1\}^n$ をキーボードから計算者が入力する操作はふつう '疑似乱数の初期化' と呼ばれている．上の議論から分かるように，初期化は計算者が '賭け' を実践するときの手に他ならないから，きわめて重要な意味を持つ．たとえば M-系列 (cf. [1, 4]) と呼ばれる疑似乱数生成法では，種が $500 \sim 10^4$ ビットに及ぶものがあり，それを人が入力することが困難なため，その種を別個の線形合同法で算出している[2]．このことは新しいモンテカルロ法の観点から見ると，'別個の線形合同法の種' こそが本当の '種' であり，様々な統計的性質は，もちろん，その初期化に用いた線形合同法をも含めて検討されるべきである．

3. 疑似乱数生成器の安全性

疑似乱数生成器 $g : \{0,1\}^n \to \{0,1\}^L$ と集合 $A \subset \{0,1\}^L$ に関して

$$|P(g(\omega') \in A) - P(\omega \in A)|$$

が十分小さいき，g は A に対して安全である，という．g が A に対して安全であれば「$g(\omega')$ に対するリスク評価は元のリスク評価と大差ない」といえるから，前節の計算者の期待をほぼ満足させることができるだろう．

注意 2. JIS[2] をはじめとして，夥しい数の疑似乱数の検定は，およそ次の手順で行われて来た:

---

[1] 疑似乱数生成器の目的からわかるように，実用のためには，当然，$n \in \mathbf{N}$ は種がキーボードから入力可能な程度に小さいことが必要である．

[2] 皮肉にも，この '別個の線形合同法' の使われ方は確かに疑似乱数生成器の目的と合致している．

(i) 検定項目 (連の検定，ポーカー検定など) を決める．

(ii) 複数の種 ω' をもとに疑似乱数生成器 g によって疑似乱数 $g(\omega')$ を生成し，その結果，棄却されるものの割り合いを計算する．

(iii) 棄却されるものの割り合いがその検定の危険率程度であれば，g を採択し，それを超えるようであれば棄却する．

上の手順において，(i) で選ばれた検定の棄却域を A とすれば，(ii) で行っていることは確率 $P(g(\omega') \in A)$ を推定する作業と解釈できる．そして (iii) ではそれが危険率 $P(\omega \in A)$ と近いかどうかを調べていることになる．従ってこうした検定作業は実は疑似乱数生成器の安全性の検査であったといえる．

当然，どんな g に対してもそれが安全でないような A が存在する．実際，たとえば A が g の値域

$$A := g(\{0,1\}^n) \subset \{0,1\}^L \tag{1}$$

の場合，

$$P(\omega \in A) \leq 2^{n-L} < P(g(\omega') \in A) = 1$$

となって，明らかに g は A に対して安全でない．そのため，すべての A を対象としていたのでは疑似乱数生成器の安全性を論ずることはできない．すなわち対象とする A のクラスを制限して疑似乱数生成器の安全性を論じなければならない．

3.1. 計算量的に安全な疑似乱数生成器

例 2 の賭けにおける集合 A は，たとえば疑似乱数生成器 $g : \{0,1\}^{100} \to \{0,1\}^{10^6}$ のある検定の棄却域 (危険率 1/32) であると考えることができる．例 2 のような集合 A の総数は 2^{10^6} 個の元の中から 2^{10^6-5} を取り出す組合せの数であるから，それをスターリングの公式を用いて下から評価すると

$$\begin{aligned} {}_{32r}C_r &= \frac{32r(32r-1)(32r-2)\times\cdots\times(32r-r+1)}{r!}, \quad r = 2^{10^6-5} \\ &> \frac{(31r)^r}{r!} \sim \frac{(31r)^r}{r^r e^{-r}\sqrt{2\pi r}} = (31e)^r/\sqrt{2\pi r} > (30e)^r > 2^{6r}. \end{aligned}$$

今，$\omega \in A$ かどうかを判定するには，そのためのプログラムを書かなければならない．A の総数は $2^{6\cdot 2^{10^6-5}}$ 個以上であるから，そのようなプログラムも同じ数だけ必要となる．各プログラムを符号化すれば，それはビット列になるが，その長さは長いものでは $6\cdot 2^{10^6-5}$ ビットに達する．なぜならば，長さが L ビットのプログラムは高々 2^L 個しかないからである．さらに，大部分の A のプログラムがほぼ $6\cdot 2^{10^6-5}$ ビットの長さに達することが容易に分かる．以上のことはプログラムの符号化の仕方 (プログラム言語) によらない．

明らかに，そのように長いプログラムは実行不可能である．言い換えると，そのように長いプログラムが必要となる検定は実際には実行できない．そこで，計算量

的に実行可能な検定にだけ採択されるような疑似乱数生成器があればそれで十分である．そのような疑似乱数生成器は計算量的に安全 (または暗号理論的に安全) である，という．

暗号理論において，計算量的に安全な疑似乱数生成器は，ここに述べた空間計算量 (プログラムの長さ) ではなく，時間計算量 (プログラムの実行時間) をもとに定式化されている．詳しくは [3, 5] を見よ．計算量的に安全な疑似乱数生成器に関する最大の問題は，それが存在するかどうか不明であることである．その存在は $\mathbf{P} \neq \mathbf{NP}$ 予想と絡む難しい問題であって厳密には未解決である．さらに，この概念は漸近的な性質を論じたものであり，個々の具体的な問題に対して，計算量的に安全な疑似乱数生成器が確実に有用であることを保証するものではない．

なお，計算量的に安全な疑似乱数生成器への確率論的アプローチが [9] で試みられている．

注意 3. M-系列 (cf. [1, 4]) では (1) の集合 A が簡単に表される．実際，その M-系列の定義式が

$$x_n := f(x_{n-p}, x_{n-p+1}, \ldots, x_{n-1})$$

の形をしている場合，$L > p$ ならば

$$A := \{\omega = (\omega_1, \omega_2, \ldots, \omega_L) \in \{0,1\}^L ; \omega_{p+1} - f(\omega_1, \omega_2, \ldots, \omega_p) = 0\}$$

である．f は簡単に計算できるので $\omega \in A$ かどうかは簡単に判定できる．このことは M-系列が計算量的に安全である疑似乱数の対極に位置することを示す．

3.2. モンテカルロ積分のために安全な疑似乱数生成器

モンテカルロ法の応用のほとんどが確率変数の数値積分 (モンテカルロ積分) であることに注目しよう．そこで Z を m 回の硬貨投げの関数，すなわち $\{0,1\}^m$ 上の関数とし，その独立なコピーの列 $\{Z_n\}_{n=1}^N$ の標本平均を X とする．X は Nm 回の硬貨投げの関数である．具体的に書けば，

$$X(\omega) := \frac{1}{N}\sum_{n=1}^{N} Z_n(\omega_n), \quad \omega_n \in \{0,1\}^m, \quad \omega = (\omega_1, \ldots, \omega_n) \in \{0,1\}^{Nm}.$$

平均は $\mathbf{E}[X] = \mathbf{E}[Z]$ であり，分散 $\mathbf{Var}[X] = \mathbf{Var}[Z]/N$ である．そこで X でもって Z の平均を推定するときのリスクをチェビシェフの不等式

$$P(|X - \mathbf{E}[Z]| > \delta) < \frac{\mathbf{Var}[Z]}{N\delta^2}$$

で評価しよう[3]．このことは，

$$A := \{\omega \in \{0,1\}^{Nm} ; |X(\omega) - \mathbf{E}[Z]| > \delta\} \tag{2}$$

[3] $\mathbf{E}[X]$ が未知であるのと同様に $\mathbf{Var}[X]$ も未知であるのが普通だろう．その意味でこのリスク評価は完全ではない．しかし状況によっては $\mathbf{Var}[X]$ の上からの評価が得られれば (たとえば X が有界の場合など)，このリスク評価は完全になる．

としたとき，$\omega \in \{0,1\}^{Nm}$ を選んで $\omega \notin A$ ならば勝ち，$\omega \in A$ ならば負け，という賭けを考えていることになる．そしてこの賭けのリスク (負ける確率 $P(\omega \in A)$) は正確には分からなくても，その上からの評価 $P(\omega \in A) < \mathbf{Var}[Z]/(N\delta^2)$ が分かる，という状況になっている．

注意 4. $\mathbf{E}[X]$ の値は未知だから A も未知であるが，確率 $P(\omega \in A)$ は評価できる．このことを賭けの比喩を用いていうと，計算者は，負ける確率の評価は知っているものの，自分が勝ったのか負けたのか，について知ることができない，ということになる．その点ではモンテカルロ積分は通常の賭けと大いに異なっている．

次の定理がある．

定理 1. (cf. [3]) $N \leq 2^m$ のとき，次の性質を満たす疑似乱数生成器 $g : \{0,1\}^{2m} \to \{0,1\}^{Nm}$ が存在する:

$$\begin{aligned} \mathbf{E}[X(g(\omega'))] &= \mathbf{E}[X(\omega)], \\ \mathbf{Var}[X(g(\omega'))] &= \mathbf{Var}[X(\omega)]. \end{aligned}$$

ただし ω と ω' はそれぞれ $\{0,1\}^{Nm}$ 上と $\{0,1\}^{2m}$ 上で一様分布するとする．

定理 1 の疑似乱数生成器 g を用いて $X(g(\omega'))$ によって $\mathbf{E}[Z]$ を推定するときのリスクは，やはりチェビシェフの不等式によって

$$P\left(|X(g(\omega')) - \mathbf{E}[Z]| > \delta\right) < \frac{\mathbf{Var}[Z]}{N\delta^2}$$

であり，(2) の集合 A に対して，$P(g(\omega') \in A) < \mathbf{Var}[Z]/(N\delta^2)$ となっている．すなわち $X(\omega)$ の場合と同一のリスク評価を持つ．この意味で，g は X のモンテカルロ積分に対して安全な疑似乱数生成器といえよう．

定理 1 の証明. g は次のように構成される: $\mathrm{GF}(2^m)$ を位数 2^m の有限体とし，任意の 2 つの全単射 $\phi : \mathrm{GF}(2^m) \to \{0,1\}^m$ と $\psi : \mathrm{GF}(2^m) \to \{1,2,\ldots,2^m\}$ をとって固定する．各 $\omega' := (x,\alpha) \in \{0,1\}^m \times \{0,1\}^m \cong \{0,1\}^{2m}$ に対して

$$Z_n(\omega') := \phi\left[\phi^{-1}x + (\psi^{-1}n)(\phi^{-1}\alpha)\right], \quad n = 1,2,\ldots,2^{2m}$$

とおき，さらに

$$g(\omega') := (Z_1(\omega'), Z_2(\omega'), \ldots, Z_N(\omega')) \in \{0,1\}^{Nm}$$

とする．このとき，ω' が $\{0,1\}^{2m}$ 上で一様分布すれば，各 $Z_n(\omega')$ は $\{0,1\}^m$ 上で一様分布し，さらに $\{Z_n(\omega')\}_{n=1}^{2^m}$ はペアごとに独立になる．実際，任意の $a,b \in \{0,1\}^m$, $1 \leq n < n' \leq 2^m$ に対して

$$\begin{aligned} &P(Z_n(\omega') = a,\ Z_{n'}(\omega') = b) \\ &\quad = P\left(\phi^{-1}x + (\psi^{-1}n)(\phi^{-1}\alpha) = \phi a,\ \phi^{-1}x + (\psi^{-1}n')(\phi^{-1}\alpha) = \phi b\right) \end{aligned}$$

ここで，未知数 (x', α') に関する $\mathrm{GF}(2^m)$ での連立1次方程式

$$\begin{cases} x' + (\psi^{-1}n)\alpha' &= \phi a \\ x' + (\psi^{-1}n')\alpha' &= \phi b \end{cases}$$

の一意解を $(x'_0, \alpha'_0) \in \mathrm{GF}(2^m) \times \mathrm{GF}(2^m)$ とすれば

$$\begin{aligned} P(Z_n(\omega') = a,\, Z_{n'}(\omega') = b) &= P\left(\{(\phi^{-1}x'_0, \phi^{-1}\alpha'_0)\}\right) \\ &= 2^{-2m} \\ &= P(Z_n(\omega') = a)P(Z_{n'}(\omega') = b) \end{aligned}$$

である．このペアごとの独立性のため定理1の主張が成り立つ． q.e.d.

注意 5. 定理1で g の定義域 $\{0,1\}^{2m}$ をもっと短いビット列の集合にすることはできない．すなわち $2m$ ビットは定理1の主張を成り立たせる最小のランダム性である．しかし，上の証明で構成した g は実際の数値計算ではプログラムが複雑になり，サンプルの生成が非常に遅いので実用的ではない．最小のランダム性ではないものの，ペアごとに独立なサンプルを高速に生成する方法が [6, 11] で開発されている．さらに，必ずしも有限回の硬貨投げの関数でないが計算可能な (ある停止時刻に関して可測であるような) 確率変数のペアごとに独立なサンプルを生成する方法も [7, 8] にある．

注意 6. ペアごとに独立なサンプリングによっても，なお，種の入力が人の能力を超える場合もある．そのような場合は，やはり汎用的な疑似乱数生成器を利用しなければならない．しかしその場合でも，もとのモンテカルロ積分に比べれば，はるかに短い疑似乱数で済むから，疑似乱数の統計的性質や生成速度にきわめて鈍感な数値積分法を提供できる．

参考文献

[1] 伏見正則，乱数，東京大学出版会，(1989).

[2] JIS Z 9031 乱数発生及びランダム化の手順，日本規格協会，2001年改正.

[3] M. Luby, Pseudorandmness and cryptographic applications, Princeton Computer Science Notes, Princeton University Press, (1996).

[4] M. Matsumoto and T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM. Trans. Model. Comput. Simul., **8-1**, (1998) 3–30.

[5] D.R. Stinson, Cryptography (Theory and practice), CRC Press, Boca Raton/ Ann Arbor / London / Washington,D.C., (1995).

[6] H. Sugita, Robust numerical integration and pairwise independent random variables, Jour. Comput. Appl. Math., **139** (2002), 1–8.

[7] H. Sugita, Dynamic random Weyl sampling for drastic reduction of randomness in Monte Carlo integration, Math. Comput. Simulation, **62** (2003), 529–537.

[8] 杉田洋, 複雑な関数の数値積分とランダムサンプリング,「数学」岩波書店 **56-1**, (2004), 掲載予定.

[9] H. Sugita, An analytic approach to secure pseudo-random generation, *peprint.*

[10] H. Sugita, The Random Sampler, 疑似乱数生成と動的ランダム-ワイル-サンプリングのための C/C++言語ライブラリ, 下記にて公開:
`http://idisk.mac.com/hiroshi_sugita/Public/imath/mathematics.html`.

[11] H. Sugita and S. Takanobu, Random Weyl sampling for robust numerical integration of complicated functions, Monte Carlo Methods and Appl., **6-1** (1999), 27–48.